Teacher.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TeacherDataManagementApp
{
    public class Teacher
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public string ClassAndSection { get; set; }

        public Teacher(int id, string name, string classAndSection)
        {
            ID = id;
            Name = name;
            ClassAndSection = classAndSection;
        }

        public override string ToString()
        {
            return $"ID: {ID}, Name: {Name}, Class and Section: {ClassAndSection}";
        }
    }
}
```

Program.cs

```csharp
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TeacherDataManagementApp
{

    internal class Program
    {
        private const string FilePath = "teachers.txt";

        static void Main(string[] args)
        {
            List<Teacher> teachers = new List<Teacher>();
            LoadTeachersFromFile(teachers);


            while (true)
            {
                Console.WriteLine("Teacher Data Management");
                Console.WriteLine("1. Add Teacher");
                Console.WriteLine("2. Display All Teachers");
                Console.WriteLine("3. Update Teacher");
            Console.WriteLine("4. Exit");

            Console.Write("Enter your choice (1/2/3/4): ");
            string choice = Console.ReadLine();

            switch (choice)
            {
                case "1":
                    AddTeacher(teachers);
                    break;
                case "2":
                    DisplayAllTeachers(teachers);
                    break;
                case "3":
                    UpdateTeacher(teachers);
                    break;
                case "4":
                    SaveTeachersToFile(teachers);
                    return;
                default:
                    Console.WriteLine("Invalid choice. Please try again.");
                    break;
            }

            Console.WriteLine();
        }
    }

    static void LoadTeachersFromFile(List<Teacher> teachers)
    {
        if (File.Exists(FilePath))
        {
            string[] lines = File.ReadAllLines(FilePath);
            foreach (string line in lines)
            {
                string[] values = line.Split(',');
                if (values.Length == 3 && int.TryParse(values[0], out int id))
                {
                    Teacher teacher = new Teacher(id, values[1], values[2]);
                    teachers.Add(teacher);
                }
            }
        }
    }

        static void AddTeacher(List<Teacher> teachers)
    {
        Console.Write("Enter Teacher ID: ");
        int id = int.Parse(Console.ReadLine());

        Console.Write("Enter Teacher Name: ");
        string name = Console.ReadLine();

        Console.Write("Enter Class and Section: ");
        string classAndSection = Console.ReadLine();

        Teacher teacher = new Teacher(id, name, classAndSection);
        teachers.Add(teacher);
```

```
            Console.WriteLine("Teacher added successfully.");
        }

        static void DisplayAllTeachers(List<Teacher> teachers)
        {
            if (teachers.Count == 0)
            {
                Console.WriteLine("No teachers found.");
                return;
            }

            foreach (var teacher in teachers)
            {
                Console.WriteLine(teacher);
            }
        }

        static void UpdateTeacher(List<Teacher> teachers)
        {
            Console.Write("Enter Teacher ID to update: ");
            int idToUpdate = int.Parse(Console.ReadLine());

            Teacher teacherToUpdate = teachers.Find(t => t.ID == idToUpdate);

            if (teacherToUpdate == null)
            {
                Console.WriteLine("Teacher with the given ID not found.");
                return;
            }

            Console.Write("Enter new Teacher Name: ");
            string newName = Console.ReadLine();

            Console.Write("Enter new Class and Section: ");
            string newClassAndSection = Console.ReadLine();

            teacherToUpdate.Name = newName;
            teacherToUpdate.ClassAndSection = newClassAndSection;

            Console.WriteLine("Teacher updated successfully.");
        }

        static void SaveTeachersToFile(List<Teacher> teachers)
        {
            using (StreamWriter writer = new StreamWriter(FilePath))
            {
                foreach (var teacher in teachers)
                {

writer.WriteLine($"{teacher.ID},{teacher.Name},{teacher.ClassAndSection}");
                }
            }
        }
    }
}
```